IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Carlos J. Gonzalez et al.		
Title:	Flash Memory System Startup Operation		
Application No.:	10/751,033	Filing Date:	December 31, 2003
Examiner:	Maskulinski, Michael C.	Group Art Unit:	2113
Docket No.:	SNDK.281US0	Conf. No.:	3703

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 C.F.R. 41.37

Further to the Notice of Appeal filed in this application on April 18, 2007, this Appeal Brief is being submitted to the Board of Patent Appeals and Interferences.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The real party in interest is SanDisk Corporation, a corporation of the state of Delaware, the assignee of all right, title and interest in the present application from the applicants Carlos J. Gonzalez and Andrew Tomlin, recorded in the United States Patent and Trademark Office at reel/frame 014807/0618 (2pages).

II. RELATED APPEALS AND INTERFERENCES

Based upon information and belief, there are no appeals or interferences that could directly affect or be directly affected by or have a bearing on the decision by the Board of Patent Appeals and Interferences in the pending appeal.

III. STATUS OF THE CLAIMS

The final rejection of claims 1-4, 10, 17, 18, 20 and 23 is being appealed. These appealed claims are reproduced in the Claims Appendix hereto.

All of original application claims 1-27 remain in the present application. The status of the claims not being appealed is as follows:

(a) Claims 21, 22 and 25-27 have been allowed, and

(b) Claims 5-9, 11-16, 19 and 24 have been objected to but indicated to be allowable if re-written in independent form.

IV. STATUS OF AMENDMENTS

No amendment has been filed since the mailing on January 19, 2007 of the final Office Action herein.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The appealed independent claims 1, 17 and 23 include a common concept, although expressed in different ways and scope. In a flash memory system that includes a processor and a random-access-memory (RAM), two copies of firmware code are stored in the non-volatile flash memory. Upon initiation of the memory system, the firmware code is transferred from the flash memory into the RAM and then executed from the RAM by the microprocessor during the course of operating the memory system.

The transfer of one firmware copy is first attempted and is completed if there are no errors in the copy or if any bit errors that exist can be corrected. But if the first copy contains more bit errors than can be corrected, at least a portion of the second copy is then transferred into the RAM instead. The goal is to provide an error-free version of the firmware in the RAM for execution by the memory system processor.

Figure 2 of the present application illustrates the address space of a portion of the flash memory in which first and second copies of the firmware code are stored (¶0029, lines 1-5). The process of uploading firmware from the flash memory into the RAM 21 (Figure 1) for execution by the system processor 19 is controlled by a small amount of boot code contain in a read-only-memory (ROM) 23 that is executed by the processor 19 (¶0030, lines 1-10) upon system initialization.

Reference is made to the flow-chart of Figure 4 of the present application, which is described in paragraphs 0033-0048 of the specification. Upon initialization 71 of the memory system by turning on the power or resetting a powered-up system (¶0033), the boot code is read at 75 and then executed by the processor 19 (¶0033, lines 6-7) to upload the firmware from the flash memory to the RAM 21.

Execution of the boot code causes a unit of data, such as a sector or a page containing multiple sectors, to be read at 81 of Figure 4 from a first copy of the firmware code (¶0037). In this example, one sector of the read firmware code is checked at a time by error-correction-code (ECC) circuits 25 (Figure 1) for any bit errors in the data read (¶0039, lines 1-4). If there are no bit errors in the sector (as determined at 83 of Figure 4), or if any bit errors can be corrected by the ECC (as determined at 91 of Figure 4 and corrected at 93), the next sector in order of the first firmware copy is identified at 85 and these steps of reading and checking are then repeated on this next sector (¶0039, lines 5-9 and ¶0040, lines 4-9). But if the number of bit errors in the sector is in excess of what can be corrected by the ECC, the processing increments to the second copy of the firmware code at 105 of Figure 4 and that sector is then read at 81 from the second copy (¶0040, lines 9-17). If this replacement sector survives the ECC checks at 83 and 91 of Figure 4, it then becomes the version of that sector that is later uploaded into the RAM 21 (Figure 1).

As is apparent from the flow chart of Figure 4, the first embodiment contains many additional processing steps but those summarized herein are believed to be the most relevant to an understanding of the claims on appeal. A second embodiment having many common processing steps is also described with respect to the flow chart of Figure 8.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The Board is being asked to review the final rejection of claims 1-4, 10, 17, 18, 20 and 23 under 35 U.S.C. § 103(a) over published United States patent applications nos. 2004/0205328 A1 (hereinafter referred to as "Langford et al.") and 2004/0088534 A1 (hereinafter referred to as "Smith et al.").

VII. ARGUMENT

Summary

The primary Langford et al. reference describes the storage of two copies of boot code in a non-volatile memory of a data processing system. But rather than check for data errors during the uploading (booting) of the boot code from that memory, as included in the appealed claims for uploading firmware into a memory system, the validity of the code is determined when it is downloaded into the memory and a flag is stored with each copy to indicate whether it is valid. When uploading the boot code during initialization of the data processing system, Langford et al. checks this flag of one of the code copies before transferring that copy from the non-volatile memory into operating memory of the data processing system. No validity check or error correction is described by Langford et al. to be performed during this uploading, contrary to the process of the appealed claims. If the flag of the first copy indicates that it is not valid, the flag of the second copy is then checked and, if it indicates that the second copy is valid, the second copy is loaded into the operating memory of the data processing system. If neither copy of the code is valid, initialization of the data processing system is aborted.

The secondary Smith et al. reference was cited in the final Office Action as evidence that it would have been obvious to modify the boot code uploading process of Langford et al. to add (or perhaps replace the validity flags with) the use of error

correction codes. It is submitted that the mere mention by Smith et al. of including an error correction code as part of a BIOS (basic input-output system) of a peripheral device (1) would not have made such a modification of Langford et al. obvious, but (2) even if, for the purpose of argument, such a combination were to be made, it would not meet the terms of the appealed claims.

The Langford et al. Reference

In Langford et al., a flag "Pside" is set when a "permanent" copy of the boot code stored in non-volatile memory is valid, and a flag "Tside" is set when a redundant "temporary" copy is valid. With reference to Figure 3, a Pside copy of the code is stored in memory 306, and a Tside copy in memory 312 (see ¶0025), along with respective Pside validity flag 320 and Tside validity flag 322.

The flow chart of Figure 4 of Langford et al. illustrates the code uploading process. When the code within the Pside memory 306 is to be used, a Pside validity flag 320 is first checked (¶¶0027,0032 and 402 of Figure 4). If valid, the Pside code is loaded into memory (¶¶0027,0032 and 404 of Figure 4). However, if the Pside code is invalid, the Tside validity flag 322 is then checked (¶¶0028, 0033 and 412, 414 of Figure 4) and the Tside code is instead loaded if valid (¶¶0028, 0033 and 416 of Figure 4). If both of the Pside and Tside flags are set to invalid, then the boot process is terminated (¶0034) because both copies of the code are likely damaged or corrupted.

The checking of the Pside and Tside flags is the only check that can be found in Langford et al. of the boot code copies during the uploading of one of them into a system memory. These flags are stored in the non-volatile memory along with their respective copies of the code to indicate whether they are valid or not. These flags are set when the boot code is loaded into the non-volatile memory or updated, as illustrated by the flow chart in Figure 5 of Langford et al. The code data is validated when loaded into the non-volatile memory (¶¶0037,0038 and 508 of Figure 5) by use of a cyclical redundancy check (CRC). The validity of the boot code is verified then instead of when being uploaded from the non-volatile memory to the system as specified in the appealed claims. When being uploaded, the validity of each of the code copies is checked only by checking their respective Pside or Tside flags.

Langford et al. clearly want to avoid beginning to load a copy of the boot code if any portion of it is invalid. "In this manner, boot time is saved from preventing the computer system from booting from a defective firmware image until an error is encountered and then having to begin the boot process again using a redundant image." (Langford et al. ¶0029, lines 1-4.) Langford et al. saves this boot time by not beginning to upload a copy of the boot code until it is determined that its stored validity flag (Pside or Tside) indicates that the data of the copy are valid.

The appealed claims, on the other hand, recite that the first firmware copy is transferred from non-volatile memory to RAM and any bit errors in the transferred copy are identified. If there are bit errors and they are correctable, they are corrected. If uncorrectable, at least some of the second firmware copy is loaded in place of at least the portion of the first copy containing the uncorrectable bit errors. There is no suggestion or mention by Langford et al. of the claimed identification and correction of erroneous boot data bits in the uploading process or reliance on the second firmware copy if the errors cannot be corrected. Loading of boot code will not be commenced unless its validity flag indicates that there are no bit errors in the stored code.

The Smith et al. Reference

The secondary Smith et al. patent appears to be cited in the final Office Action solely for its mention of use of error correction codes, with only paragraph 0047 of Smith et al. being referenced. This paragraph merely mentions that when BIOS (basic input-output system) code is updated, a corresponding checksum or error correction code is also updated. The same disclosure is repeated in paragraph 0052 (lines 11-14) of Smith et al. Other than those two mentions, nothing can be found in Smith et al. that discusses checksums or error correction codes or their use.

Langford et al. and Smith et al. Would Not Have Made the Appealed Claims Obvious

It is submitted that Smith et al. would not have made it obvious to correct erroneous bits of the first copy of the boot code of Langford et al. with an error correction code as the boot code is being transferred out of non-volatile memory into RAM. Nor would Smith et al. have made it obvious to then, if the first copy cannot be corrected, to load at least some portion of the second copy. Smith et al. merely mentions that it is desirable to update a checksum or error correction code when the code upon which it is

based is updated. It is not seen how this could have suggested any modification of Langford et al., let alone the modifications necessary to meet the terms of the appealed claims.

It is further submitted that Langford et al. expressly teach against making any such modification by stressing the importance of determining the validity of a boot code copy before any loading of it into system RAM is allowed to begin. "The mechanism of the present invention saves on boot time by preventing a computer system from booting from a defective image and then having to switch to another image after the defective portion of the defective image has been encountered." (Langford et al., ¶0040, lines 5-9.) The appealed claims define the opposite, wherein the transfer of a copy of the firmware begins and is then checked for errors. Langford et al. teach against reading its boot code without first knowing that it is valid by checking its Pside or Tside flag that is stored along with the code.

The appealed claims also specify that if any errors in the transfer of one copy of the code can be corrected, they are corrected, but if not, at least some of the second copy of the code is transferred instead. Langford et al. certainly do not suggest switching from transferring data from one code copy to the other. To the contrary, Langford et al. stress not having to switch between two copies, as quoted in the preceding paragraph. Smith et al. would certainly not have suggested, by its mere mention of updating error correction codes, that the operation of Langford et al. should be modified in a manner contrary to its expressed goals.

Conclusion

For the reasons given above, it is respectfully submitted that the appealed claims are patentable over the cited Langford et al. and Smith et al. references. A decision of the

Board reversing the final rejection of the Examiner, with instructions that the present application be allowed, is solicited.

FILED VIA EFS

Respectfully submitted,

Gerald P. Parsons — August 17, 2007
Reg. No. 24,486 — Date

Davis Wright Tremaine LLP
505 Montgomery Street, Suite 800
San Francisco, CA 94111-6533
(415) 276-6500 (main)
(415) 276-6534 (direct)
(415) 276-6599 (fax)

VIII. CLAIMS APPENDIX

The following is the text of the appealed claims:

1. A method of initiating a memory storage system having flash memory containing at least first and second copies of firmware code stored in different locations therein, a microprocessor, a read-only-memory (ROM) containing microprocessor accessible boot code and a random-access-memory (RAM) for storing microprocessor accessible firmware code, the method comprising:

executing the boot code to transfer a first copy of the firmware from the flash memory to the RAM,

identifying any bit errors in the transferred first copy of the firmware code,

if bit errors are identified that are correctable, correcting the erroneous bits,

if bit errors are identified that are not correctable, reading at least a portion of the second copy of the firmware code into the RAM in place of at least a portion of the first copy containing the uncorrectable bit errors, and

executing an error free copy of the firmware code from the RAM.

2. The method of claim 1, wherein identifying any bit errors in the transferred first copy includes calculating error-correction-codes (ECCs) from individual portions of the first copy of the firmware by passing the firmware portions through ECC circuitry in succession as they are being transferred from the flash memory to the RAM, and comparing the calculated ECCs with ECCs previously calculated from said portions of the first copy of the firmware data.

3. The method of claim 2, wherein correcting the erroneous bits includes the microprocessor executing an error correction algorithm of the boot code to correct erroneous bits.

4. The method of claim 2, wherein the individual portions of the first copy of the firmware code include one or more sectors of data and an ECC previously calculated therefrom and stored in the flash memory therewith.

(Claims 5-9 objected to but indicated to be allowable if rewritten.)

10. The method of claim 1, additionally comprising, prior to executing the boot code to transfer a first copy of the firmware from the flash memory to the RAM, checking the state of a firmware present flag that is set when firmware is stored in the flash memory and continuing to execute the boot code to transfer the first copy of the firmware from the flash memory to the RAM only when the firmware present flag is set.

(Claims 11-16 objected to but indicated to be allowable if rewritten.)

17. A method of operating a memory storage system having flash memory, a microprocessor, a read-only-memory (ROM) containing boot code accessible by the microprocessor, a random-access-memory (RAM) and circuitry that calculates an error correction code (ECC) from data passing through it, the method comprising:

storing at least first and second copies of firmware code in different addressable locations of the flash memory by passing the firmware copies one at a time through the ECC circuitry and storing the ECCs calculated thereby in the flash memory,

thereafter initiating operation of the memory system by causing the microprocessor to execute the boot code to transfer the first copy of the firmware from the flash memory to the RAM through the ECC circuitry which calculates an ECC therefrom,

utilizing the calculated and stored ECCs to identify any bit errors in the transferred first copy of the firmware code, and

if bit errors are identified to be correctable, causing the microprocessor to execute an error correction algorithm within the boot code to correct the erroneous bits, in order to result in the firmware code being loaded into the RAM without any errors, or

if bit errors are identified to be uncorrectable, transferring at least a portion of the second copy of the firmware code into the RAM in place of at least a portion of the first copy containing the uncorrectable bit errors, in order to result in the firmware code being loaded into the RAM without any errors.

18. The method of claim 17, wherein storing the firmware code includes storing ECCs individually calculated from one or more sectors of the firmware code.

(Claim 19 objected to but indicated to be allowable if rewritten.)

20. The method of claims 17, wherein storing the firmware code additionally includes setting a flag to indicate the presence within the flash memory of at least one firmware copy, and wherein executing the boot code to transfer either of the first or second copies of the firmware code includes first reading the flag associated therewith and proceeding to read the copy of the firmware code only if the associated flag is set.

(Claims 22 and 23 have been allowed.)

23. A flash memory storage system, comprising:

an array of flash memory cells storing data in charge storage elements and containing at least first and second copies of firmware code stored therein along with respective first and second sets of error-correction codes (ECCs) calculated from the first and second copies of the firmware code,

a controller processor,

circuitry that calculates ECCs from data passing through the circuitry,

a read-only-memory containing boot code that the processor accesses and executes in response to initialization of the storage system,

a random-access-memory that is accessible by the processor to obtain instructions to be executed, and

wherein the boot code causes the processor to read the first firmware code copy including passing the read first firmware code copy through the ECC calculation circuitry

which calculates ECCs and provides with respect to the first set of ECCs stored with the first firmware code copy a status with respect to any data errors existing in portions of the first firmware code copy to which the ECCs pertain, and

(A) if the status indicates that there are no data errors in a given one of the portions of the first firmware code copy, thereafter writing the given portion of the first copy of the firmware code into the random-access-memory, but

(B) if the status indicates that there are data errors in the given portion of the first firmware code copy, the boot code causes the processor to determine whether the number of bit errors in the firmware code exceed a given number, and

(i) if the number of bit errors do not exceed the given number, further causes the processor to correct the erroneous bits and write the corrected first firmware code copy into the random-access-memory, but

(ii) if the number of bit errors is equal to or exceeds the given number, further causes the processor to read at least a portion of the second firmware copy, pass the read second firmware code through the ECC calculation circuitry which calculates at least one ECC therefrom and provides a status with respect to any data errors existing in said at least a portion of the second firmware code copy to which said at least one ECC pertains, and if the status indicates that there are no data errors in said at least one portion of the second firmware code copy, thereafter writing said at least one portion of the read second copy of the firmware code into the random-access-memory.

(Claim 24 is objected to but indicated to be allowable if rewritten.)

(Claims 25-27 have been allowed.)

IX. EVIDENCE APPENDIX

None

X. RELATED PROCEEDINGS APPENDIX

None